ソフトウエア分散共有メモリSMS
―UDPによる実装―

緑川 博子　大橋 祐介　菊池康祐　飯塚 肇
成蹊大学 工学部

ユーザレベルソフトウエアによる分散共有メモリシステムSMSにおいて、従来のTCP通信に加え、新たにUDPによる通信を可能にし、用いる通信路の条件に応じユーザが選べるようにした。さらに、条件変数関数、ページ変更情報diffの粒度制御機構などを導入し、プログラムユーザインタフェースの向上、通信量の削減を行った。代表的なソフトウエア共有メモリシステムであるTreadMarksやメッセージパッシングライブラリのMPICH、PVMとの比較も行った。

Software Distributed Shared Memory SMS
- UDP implementation -

Midorikawa H., Ohashi Y., Kikuchi K., Iizuka H.
Seikei University

The SMS, a software distributed shared memory system, has been implemented with UDP. In addition, it now supports condition variables to make parallel programming more flexible. A technique of granularity-based page comparison-and-update achieved much reduction in the amount of communication for page update. Performance comparisons with PVI, MPI as well as the representative software DSM, TreadMarks are shown.

１．はじめに

筆者らは、OSに手を加えたり、特殊な通信機構を前提とすることなしに、ユーザレベルのソフトウエアだけで容易に構築できるソフトウエア共有メモリシステムSMSを提案、実装してきた[1][2]。SMSは、Linux/FreeBSD上にTCPにより実装されていたが、従来のTCPに加え、UDPをサポートし、ユーザが通信デバイスに応じて選択できるようにした。さらにメモリコンシステンシ維持のための差分更新情報に共有変数の型を考慮した粒度制御を加え、差分情報の通信量の低減を図った。また条件変数用の関数を新たに加え、共有変数の排他制御をより柔軟に行えるようにした。

２．ソフトウエア共有メモリシステムSMS

SMSは以下のような特徴を持つ。

- **ユーザレベルソフトウエアによる実装**
- **ネットワークを選ばないTCP/UDP通信**
  特殊な通信機能を用いず、イーサ、Myrinetなど TCPもしくはUDPをサポートする通信媒体であれば、どのようなものでも使用することができる。
- **ハードウエア独立な柔軟仮想プロセス実行**
  単一CPUから、複数CPUのPCクラスタまで、ハードウエア上のPC数、CPU数に独立にプログラムを作成することが可能。
- **新しい緩和型メモリコンシステンシモデル**
  暗黙バインディングエントリコンシステンシモデルIBECの提案、採用した。従来のエントリコンシステンシとは異なり、共有データ変数と同期変数との関連付けをプログラム中で明示する必要がない。

今回UDPによる実装の他に以下の機能追加を行った。

- **条件変数関数の導入**
  sms_cond_signal(),sms_cond_wait()などの条件変数関数を導入し、排他制御の必要な共有変数で、その共有変数の値の変化などを知らせることが可能になり、無駄なポーリングや、その共有変数をアクセスする他のプロセスへの悪影響を排除できる。
- **差分情報（diff）の粒度毎の比較・適用**
  従来のsms_alloc()に加え、sms_calloc()関数を実装し、引数から得られる各共有変数の構成データ型単位で、メモリの一貫性を保つためのページ差分データ(diff)作成、比較、適用を行う。ユーザからは連続データを変更したように見えても（例えば配列要素すべてに対する数値変化の少ない計算）、実際にはデータ内の一部分のバイトしか変化しないため、１バイト単位の比較では、変化部分が不連続バイトであるため、大量のdiffが発生するという状況があった。しかし、粒度情報を用いて比較することにより、連続データの変更は、一つのdiffとして生成され、メッセージ数、およびヘッダーを考慮すると、総通信バイト数も低減できることがわかった。さらにdiff適用時の処理時間も低減できる。

SMSでサポートする関数、定数を表１に示す。

３．性能評価結果

本実験は表２の２つの環境で評価を行った。

```
sms_nproc   ：並列稼動プロセス数
sms_proc_id  ：プロセスID(0,1,2,...)
void sms_startup(int argc,char *argv)  初期化関数
void sms_shutdown()  正常終了関数
void sms_error(char *errmsg)  エラー終了関数
void *sms_alloc(int size,int PageM_pid) 共有データ割付関数
void *sms_calloc(int num, int itemsize,int PageM_pid)
vois   sms_change_page_manager(void *adr,int size,int PageM_pid)
                                              ページマネジャ変更関数
void sms_barrier(int BarrierM_pid)  バリア同期関数
void sms_lock(int Lock_id)  ロック獲得関数
void sms_unlock(int Lock_id)  ロック解放関数
void sms_cond_wait(int cond_id, int Lock_id) 条件待ち関数
void sms_cond_signal(int cond_id)    条件成立通知関数
void sms_cond_broadcast(int cond_id) 条件成立放送関数
```

表１　SMSの定数及び関数

環境1　CPU:Celeron 400MHz メモリ:128MB OS:　RedHat Linux 6.0 環境2 CPU:MMXPentium 166MHzメモリ:64MB OS:　FreeBSD2.2.2R ネットワーク　100Mbpsイーサネット

表２　評価システムの環境

3.1　差分情報粒度制御の効果

FFTにおけるdiffの粒度制御の効果を図１に示す。callocを用いると整数のみならず浮動小数データのdiff量軽減にも有効であることが多い。

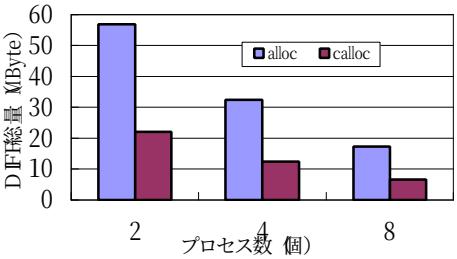

図１　FFTにおけるdiff総量(MB)

3.2　メッセージパッシングシステムとの比較

図２と図３に1000個の多体問題とマンデブロー集合計算を行ったの場合の環境１における各方式の速度向上比を示す。代表的なMPICHとPVMとの比較しても、遜色のないことがわかる。

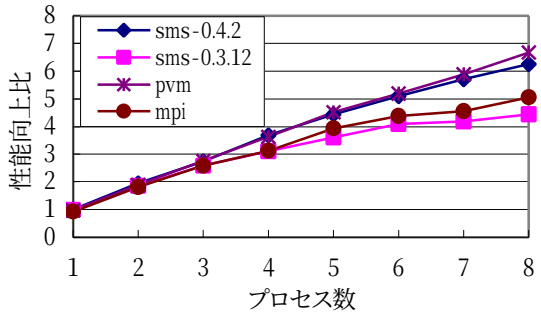

図２多体問題（1000体,100回）

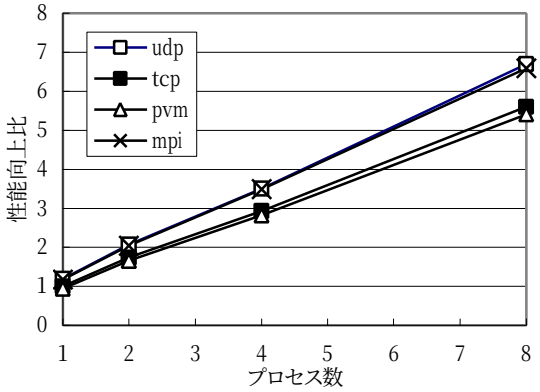

図３マンデルブロー集合計算(サイズ4096^2)

3.2　TreadMarksとの比較

環境２において上記２種の問題を代表的なソフトウエア共有メモリシステムTreadMarksとSMS-UDPを比較した結果を図４図５に示す。同等以上の性能が得られている。

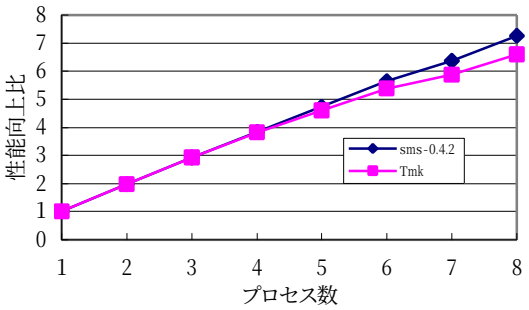

図４　多体問題（1000体,100回）

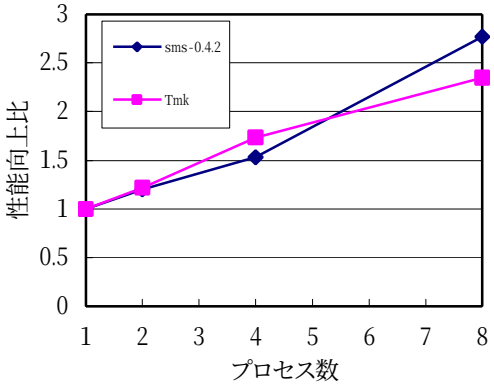

図５マンデルブロー集合計算(サイズ1024^2)

参考文献

[1] 大橋，緑川，飯塚:"PCクラスタにおける共有メモリの実装(3)－UDPによる実装－",情処全国大会 5Q08(2001,3)

[2]緑川，伊藤，大橋，飯塚:"PCクラスタにおけるユーザレベルソフトウエア分散共有メモリSMS",並列処理シンポジウムJSPP'00　論文集 p.165 (2000,5)